研究報告

メニーコア時代の超並列計算手法研究とその応用

計算機センター支援組織　助教　鏑　木　崇　史
計算機センター支援組織　助教　勝　野　弘　康

2005 年頃まで演算装置の処理能力は、半導体素子を微細化することで順調に進化し続けていたが、電気信号的特性・微細化技術などの要因で単体での処理能力向上が望めない状況にあった。その問題を打開するために、複数の演算装置（CPU）を合わせ持つマルチコア計算機が登場し、現在ではパソコンにも採用されるほど一般的なものとなった。この流れはさらに加速しており、単独の演算装置はさほど高性能ではないが、多数集めることにより高速化を図る手段をメニーコアと呼び、マルチコアの次世代技術とされている。これらのメニーコア技術は CPU だけではなく、詳細な 3D 描画のために高速化が著しく発展したグラフィックプロセッサ（GPU）にも適用されている。このような GPU を 3D 描画ではなく、一般的な計算に用いる方法を GPGPU（General-purpose computing on graphics processing units）とよび、現在研究が進められている。実際、東京工業大学の TSUBAME などのようにこのようなメニーコア技術を利用したスーパーコンピュータも数多く登場している。

これらのメニーコア演算装置を有効活用するには、ハードウェアの対応のみならず、ソフトウェアも同時に対応する必要がある。しかしながら、伝統的なプログラムの多くは並列計算されることを想定しておらず、特に計算速度が必要とされる科学技術計算分野においてはその対応が急務である。

本研究では、超並列計算時代においてメニーコア演算装置を有効活用する手法を確立することを目標として研究を推進している。具体的には次の 4 項目を目標としている。

1）マルチコア・メニーコアの計算機アーキテクチャの検討
2）評価データによる速度比較
3）実問題への適用手法の確立
4）マルチコア・メニーコアに適した計算手法と対象の検討

1）マルチコア・メニーコアの計算機アーキテクチャの検討

比較的廉価に購入可能なマルチコア型の計算機に高性能 GPU を実装し、マルチコア・メニーコアアーキテクチャの比較を実施する環境を構築した。本研究では、下記二つのシステムを構築して性能評価を行った。

S1:Intel Core i7 2600K（4 コア　3.4GHz）+ nVIDIA Quadro 4000
S2:Intel Core i7 3930K（12 コア　3.2GHz）+ nVIDIA GeForce GTX560

次節からの検証では下記表のような環境で評価を行っている。

2）評価データによる計算速度比較	S1
3）i）分子動力学法を用いた希ガス元素の結晶成長に関する研究	
3）ii）隠れマルコフモデルを用いた生命情報処理に関する研究	S2

コンパイラについては、標準的に用いられている GNU Compiler Collection(gcc)、Intel CPU に特化したコンパイラである Intel C　Compiler(icc)、GPGPU 利用のために提供されている NVIDIA 社の CUDA compiler(nvcc) の三つを想定する。

2）評価データによる計算速度比較

倍精度実数型の行列積計算を行い、CPU と GPGPU での行列サイズの違いによる計算時間比較を行った。その結果を下記に記す。

コンパイラ等の違い		icc	nvcc	nvcc（shared）
行列サイズ	1024x1024	0.58[s]	0.41[s]	0.05[s]
	2048x2048	7.48[s]	3.15[s]	0.40[s]
	4096x4096	60.18[s]	ー	3.15[s]

GPGPU の共有メモリ（GPU の L1 キャッシュ）を明示的にうまく使うことで CPU よりも 20 倍程度の性能が出せることが分かった。逆に共有メモリをうまく使えないと、性能を引き出せず、icc と同程度の性能しか出ない。

また､GPGPU を利用するにあたって思わぬ障害が一点現れた｡今回用いた計算機環境においては、GPU カードは 1 枚のみである。計算と実際の描画の制御を 1 枚のカードで行うためか、GPGPU での処理を計算が 10 秒程度占有してしまうと、プロセスが止められてしまう。行列サイズ 4096x4096 において共有メモリを使用しない nvcc の計算では、おそらくこの制限にひっかかったため、計算が正常に終了できなかった。これを回避するためには、GPU カードを 2 枚にし、計算するカードでは描画を行わないようにすればよいようである。なお、計算が途中で止まるという挙動については OS に依存するようであることを附記しておく。

3）実問題への適用手法の確立

申請者らがこれまでに作成してきた科学技術計算コードを、メニーコアに対応したプログラムコードに改変し、その性能評価を行った。具体的には以下の 2 テーマを対象とした。

i）分子動力学法を用いた希ガス元素の結晶成長に関する研究。

ii）隠れマルコフモデルを用いた生命情報処理に関する研究。

i）分子動力学法を用いた希ガス元素の結晶成長に関する研究

結晶が気相成長するとき、粒子は気相から一つずつ固相へと移っていく。その様子を調べる方法の一つとして分子動力学法がある。粒子の運動をニュートン方程式で記述し、多粒子系を数値的に扱う。系の相転移温度を調べられる他、系の緩和の様子を調べることで、新しい結晶成長描像を得ることに役立つ。近年ではガラス転移など、実験的には観察が難しいような検証などにも用いられている。この研究では非常にたくさんの粒子の運動方程式を解く必要があり、非常に計算時間がかかる。しかしながら、各粒子の運動方程式を GPGPU のもつメニーコアに任せることで計算時間の短縮が望める。

3 次元空間の周期境界条件の中にある 2048 個の粒子の運動を計算した。粒子間ポテンシャルは標準的に使われるレナードジョーンズポテンシャルを採用した。なお、倍精度計算をおこなったため、5 桁の精度で CPU の計算結果と GPU の計算結果が一致している。各粒子の運動方程式を 100 回解くのにかかる時間を次の表に示す。経過時間は 10 回の平均値である。

コンパイラ	gcc	icc	nvcc(shared)
平均経過時間 [s]	13.6	3.27	0.55

本研究で用いた分子動力学法においては、標準的なコンパイラである gcc に比べ、Intel CPU に特化したコンパイラである icc のほうが 4.15（=13.6/3.27）倍の性能向上がみられた。ここではプログラムに一切の変更を加えていない。さらに GPGPU 用にソースコードを改変すると、5.9（=3.27/0.55）倍の性能向上がみられた。gcc 利用に比べれば、nvcc 利用は 25（=13.6/0.55）倍の性能向上が得られたことになる。

ii）隠れマルコフモデルを用いた生命情報処理に関する研究

この研究ではタンパク質のアミノ酸配列を隠れマルコフモデル（HMM）でモデル化する。HMM は確率的な枠組みであり、多量の行列演算を行うため、並列化の恩恵を受けることができると予想した。HMMER というオープンソフトを入手し、その GPGPU 版である CUDA HMMER と 537,505 配列を処理するのにかかった時間を比較した。CPU を用いた計算が 303.49 秒だったのに対し、GPGPU を用いた場合は 125.26 秒となった。しかし、CPU のみを使う手法は 1 スレッドの場合であり、12 スレッドを使い切る設定では 4.75 秒とより高速になる。こちらの例では GPGPU の恩恵を受けることができなかった。これは、GPGPU の宿命とも言えるメモリ転送におけるボトルネックであると思われる。本課題で用いるデータは、アミノ酸配列数で 537,505 配列と大量になり、テキストで 2.5GB のデータ容量にもなる。GPGPU で演算を行うにはデータを通常のメモリから GPU

上のメモリに転送する必要があり、その帯域はメモリに比べ極端に狭い。このため、大量のデータを GPU 上に転送する必要のある計算には不利に働く。

一方、CPU で実行するプログラムに関してはコンパイラの種類を gcc から icc に変更し、対象のコアに特化した最適化を行うことで、30%程度の速度向上が観測できた。このことから、icc などの自動ベクトル化機能を有するコンパイラによる最適化は本研究課題においては有効であると確認された。

4）マルチコア・メニーコアに適した計算手法と対象の検討

前述のとおり、ある課題においてはメニーコアシステムが有利に働き、一方で別の課題においては不利に働く結果が得られた。このことから、対象とする課題に取り組む際にどちらが有利かを見極める必要がある。今回はどちらも行列計算を有する課題であったので、どちらの課題においても性能向上が期待されたが、実際は後者の課題においては性能が向上しなかった。これはメモリ転送のボトルネックが影響していることが予想される。このため、より一般的なプログラムを GPGPU で実施するにはプログラム内のメモリアクセスを監視し、転送に見合う効率向上が見込めるかを判断する仕組みが必要になると予想される。

Intel Compiler 利用はほとんどの場合、ソースコードの変更なく利用できる。特に近年ではソース解析がうまくなされており、自動ベクトル化、自動並列化などの機能が強力に働く。ユーザとしては計算方法を単純なソースコードに書くだけでよく、非常に利用しやすいと言える。一方、GPGPU 利用については、若干のソースコード修正が必要である。また、非常にトリッキーな仕様になっており、ユーザの利便性という点ではまだまだ不十分な点が多い。しかしながら本研究でも見られたように、GPU の仕様を理解し、共有メモリを利用するだけでも icc 利用に比べて 6 倍の性能向上につながる。ハードウェアをさらに知ることで性能を存分に引き出すことは可能であると考える。ソフトウェアであるコンパイラやハードウェアである GPU の性能は今も発展途上である。シングルスレッドの CPU 計算においてハードウェア、ソフトウェアが大幅に発展してきたように、GPGPU についても同様のことが期待される。今後の発展に伴って、GPGPU 利用による計算速度の向上を狙うメリットは非常に大きくなっていくと考えられる。

GPGPU に限らず、メニーコアシステム全体についても発展途上である。2012 年末には Intel からメニーコアシステムである Intel Phi が発表された。これは GPU ではなく、通常の CPU と同様の命令が実行できるコアが 50 個程度搭載されたシステムになっている。このような新しいハードウェアの登場により、メモリのホットスポットを事前にロードするなどのキャッシュコントロールに似た仕組みが実装されればメモリ転送の問題は大きく解消できると考えられる。